Brian Terrell
06/13/2017
Final: Space Game

Space Game Design Document – Design Notes

This project required the design of a game with multiple rooms where a character is expected to traverse the rooms collecting the necessary Items required to complete the game. In the case of this game, my goal was for the character to find the items required to kill an alien on the ISS. Below are some of the early design considerations that went into this project.

Prompt user to enter the control room and search for items or navigate to another room:

- Navigation menu is part of each child class and returns an int to the playGame() function of the game class.
- If search is chosen, the search function of the object class is called.
- If another room is chosen, the current location pointer is updated and the playGame() function loops and calls the new location's navigation function.

Navigate to another room:

- If the room has a gate keeper, it is called (roomChallenge()) and the user must complete a puzzle, or have a certain item.
- Items are stored in a vector of strings, which belongs to the Purse class.
- Purse class belongs to the Game class and is passed to the Room derived class function by pointer

Prompt user to choose area of room to search:

- Search menu calls one of the search area functions of the child class. It receives the Game class Purse object the by pointer.
- Some objects are simple and can be added to the Purse vector with no other conditions, and some require a condition to be met.
- User is given the option to pick up item, if yes then truth is passed back to the search menu function
- If the search menu function receives a true from the search are function then a flag is set to mark the item is picked up
- User is can continue to search or escape back one level

Pseudocode for game driver:

```
Set start time to current time
Loop till win, die, or exit
    Print time remaining
    Print navigation menu for starting/current location
        Options:
            Call search function
            Set current location to the room above and break
            Set current location to the room below and break
            Set current location to the room left and break
            Set current location to the room right and break
    Update current time
    Print time remaining
    Check if play should loop again
```

Pseudocode for search menu:

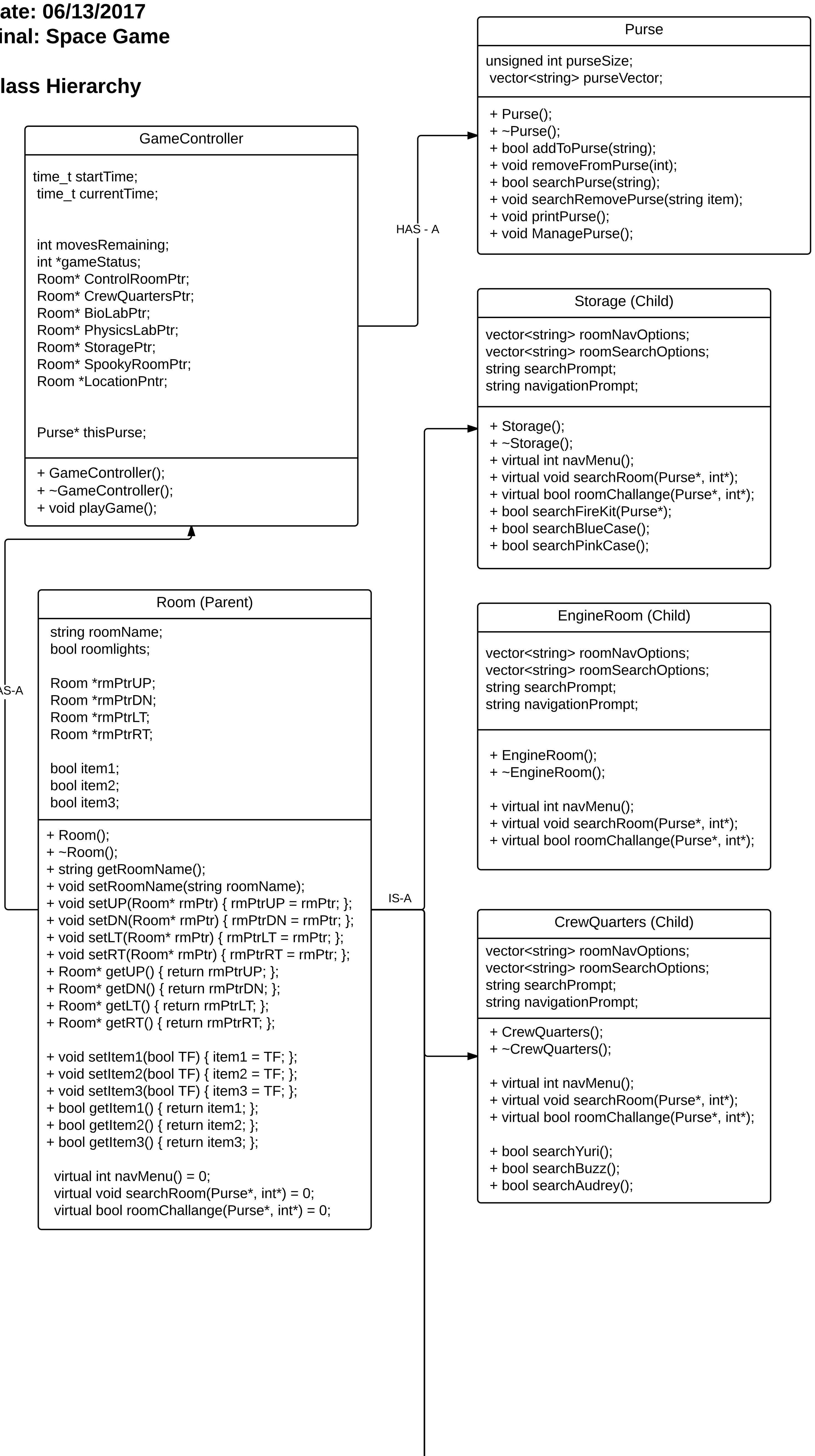
```
Call room challenge (if it has one)
Loop till exit
    Print search menu current location
        Options:
            Call navigation menu
            Call search area 1 function
                If above function returns “true”, add item to purse and set item as found
            Call search area 2 function
                If above function returns “true”, add item to purse and set item as found
            Call search area 3 function
                If above function returns “true”, add item to purse and set item as found
            Call mange Purse function
```

Pseudocode for search area function:

```
Check if item has been picked up
Check if there is a conditional to picking up this item and if it has been met, print status
Print pick up prompt
Receive input if user wants to pick up item
    If yes, then return true, else return false
```

Name: Brian Terrell
Date: 06/13/2017
Final: Space Game

Class Hierarchy



```
Purse
-----
unsigned int purseSize;
 vector<string> purseVector;
-----
 + Purse();
 + ~Purse();
 + bool addToPurse(string);
 + void removeFromPurse(int);
 + bool searchPurse(string);
 + void searchRemovePurse(string item);
 + void printPurse();
 + void ManagePurse();
```

```
GameController
-----
time_t startTime;
 time_t currentTime;


 int movesRemaining;
 int *gameStatus;
 Room* ControlRoomPtr;
 Room* CrewQuartersPtr;
 Room* BioLabPtr;
 Room* PhysicsLabPtr;
 Room* StoragePtr;
 Room* SpookyRoomPtr;
 Room *LocationPntr;


 Purse* thisPurse;
-----
 + GameController();
 + ~GameController();
 + void playGame();
```

```
Storage (Child)
-----
vector<string> roomNavOptions;
vector<string> roomSearchOptions;
string searchPrompt;
string navigationPrompt;
-----
 + Storage();
 + ~Storage();
 + virtual int navMenu();
 + virtual void searchRoom(Purse*, int*);
 + virtual bool roomChallange(Purse*, int*);
 + bool searchFireKit(Purse*);
 + bool searchBlueCase();
 + bool searchPinkCase();
```

```
Room (Parent)
-----
 string roomName;
 bool roomlights;

 Room *rmPtrUP;
 Room *rmPtrDN;
 Room *rmPtrLT;
 Room *rmPtrRT;

 bool item1;
 bool item2;
 bool item3;
-----
+ Room();
+ ~Room();
+ string getRoomName();
+ void setRoomName(string roomName);
+ void setUP(Room* rmPtr) { rmPtrUP = rmPtr; };
+ void setDN(Room* rmPtr) { rmPtrDN = rmPtr; };
+ void setLT(Room* rmPtr) { rmPtrLT = rmPtr; };
+ void setRT(Room* rmPtr) { rmPtrRT = rmPtr; };
+ Room* getUP() { return rmPtrUP; };
+ Room* getDN() { return rmPtrDN; };
+ Room* getLT() { return rmPtrLT; };
+ Room* getRT() { return rmPtrRT; };

+ void setItem1(bool TF) { item1 = TF; };
+ void setItem2(bool TF) { item2 = TF; };
+ void setItem3(bool TF) { item3 = TF; };
+ bool getItem1() { return item1; };
+ bool getItem2() { return item2; };
+ bool getItem3() { return item3; };

  virtual int navMenu() = 0;
  virtual void searchRoom(Purse*, int*) = 0;
  virtual bool roomChallange(Purse*, int*) = 0;
```

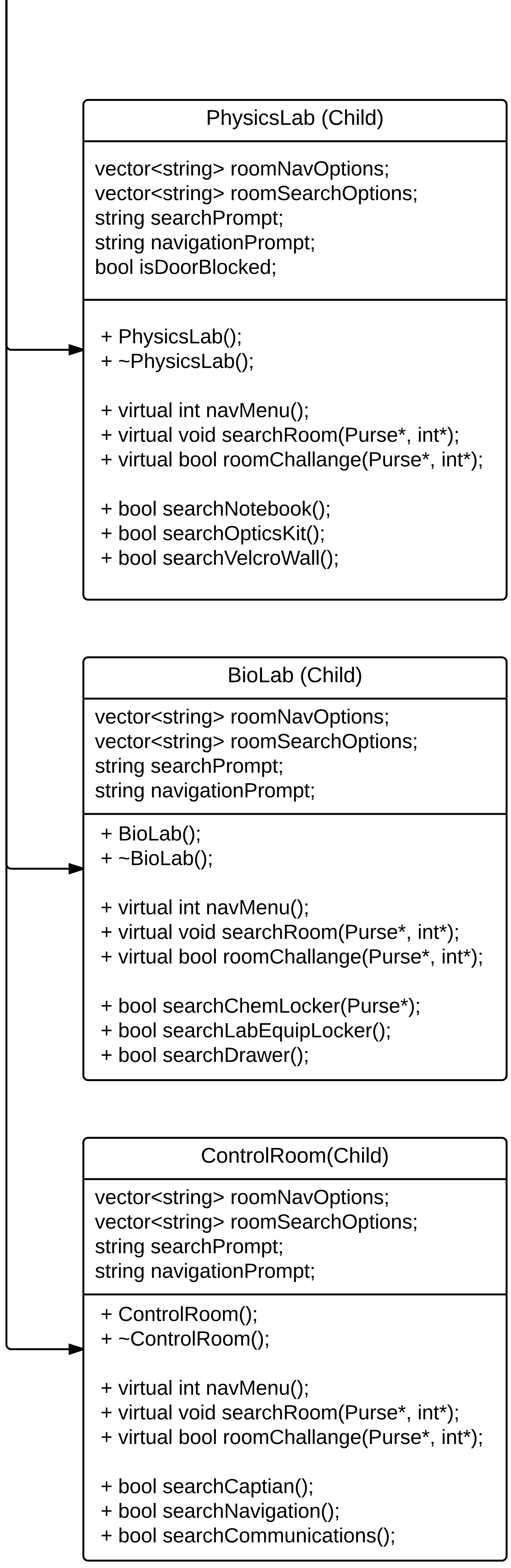
```
EngineRoom (Child)
-----
vector<string> roomNavOptions;
vector<string> roomSearchOptions;
string searchPrompt;
string navigationPrompt;
-----
 + EngineRoom();
 + ~EngineRoom();

 + virtual int navMenu();
 + virtual void searchRoom(Purse*, int*);
 + virtual bool roomChallange(Purse*, int*);
```

```
CrewQuarters (Child)
-----
vector<string> roomNavOptions;
vector<string> roomSearchOptions;
string searchPrompt;
string navigationPrompt;
-----
 + CrewQuarters();
 + ~CrewQuarters();

 + virtual int navMenu();
 + virtual void searchRoom(Purse*, int*);
 + virtual bool roomChallange(Purse*, int*);

 + bool searchYuri();
 + bool searchBuzz();
 + bool searchAudrey();
```



```
PhysicsLab (Child)
------------------------------------------------
vector<string> roomNavOptions;
vector<string> roomSearchOptions;
string searchPrompt;
string navigationPrompt;
bool isDoorBlocked;
------------------------------------------------
 + PhysicsLab();
 + ~PhysicsLab();

 + virtual int navMenu();
 + virtual void searchRoom(Purse*, int*);
 + virtual bool roomChallange(Purse*, int*);

 + bool searchNotebook();
 + bool searchOpticsKit();
 + bool searchVelcroWall();
```

```
BioLab (Child)
------------------------------------------------
vector<string> roomNavOptions;
vector<string> roomSearchOptions;
string searchPrompt;
string navigationPrompt;
------------------------------------------------
 + BioLab();
 + ~BioLab();

 + virtual int navMenu();
 + virtual void searchRoom(Purse*, int*);
 + virtual bool roomChallange(Purse*, int*);

 + bool searchChemLocker(Purse*);
 + bool searchLabEquipLocker();
 + bool searchDrawer();
```

```
ControlRoom(Child)
------------------------------------------------
vector<string> roomNavOptions;
vector<string> roomSearchOptions;
string searchPrompt;
string navigationPrompt;
------------------------------------------------
 + ControlRoom();
 + ~ControlRoom();

 + virtual int navMenu();
 + virtual void searchRoom(Purse*, int*);
 + virtual bool roomChallange(Purse*, int*);

 + bool searchCaptian();
 + bool searchNavigation();
 + bool searchCommunications();
```

Brian Terrell
06/13/2017
Final: Space Station Game

Space Ship Game Design Document - Test Plan

Menu Function

Test Case	Input Values	Driver Functions / Files	Expected Outcomes	Observed Outcomes
Navigation Menu choice: Input too low	Input < 0	Menu.cpp, main.cpp, validator.cpp, random.cpp, BioLab.cpp, ControlRoom.cpp, CrewQuarters.cpp, EngineRoom.cpp, GameController.cpp, PhysicsLab.cpp, Purse.cpp, Room.cpp, Storage.cpp	Prompt user to enter valid value, loops until satisfied.	As expected
Navigation Menu choice: Input boundary low	Input = 0		Accepts value and continues to next prompt, produces correct size board	As expected
Navigation Menu choice: Input correct range	0 <= Input <= maxListed		Accepts value and continues to next prompt	As expected
Navigation Menu choice: Input too high	Input > maxListed		Accepts value and continues to next prompt, produces correct size board	As expected
Navigation Menu choice: Input boundary high	Input = maxListed		Accepts value and continues to next prompt	As expected
Search Menu choice: Input too low	Input < 0	Menu.cpp, main.cpp, validator.cpp, random.cpp, BioLab.cpp, ControlRoom.cpp, CrewQuarters.cpp, EngineRoom.cpp, GameController.cpp, PhysicsLab.cpp, Purse.cpp, Room.cpp, Storage.cpp	Prompt user to enter valid value, loops until satisfied.	As expected
Search Menu choice: Input boundary low	Input = 0		Accepts value and continues to next prompt, completes correct number of steps	As expected
Search Menu choice: Input correct range	0 <= Input <= maxListed		Accepts value and continues to next prompt, completes correct number of steps	As expected
Search Menu choice: Input too high	Input > maxListed		Prompt user to enter valid value, loops until satisfied.	As expected
Search Menu choice: Input boundary high	Input = maxListed		Accepts value and continues to next prompt, completes correct number of steps	As expected
Main Menu: Input too low	Input < 0	Menu.cpp, main.cpp, validator.cpp, random.cpp, BioLab.cpp, ControlRoom.cpp, CrewQuarters.cpp, EngineRoom.cpp, GameController.cpp, PhysicsLab.cpp, Purse.cpp, Room.cpp, Storage.cpp	Prompt user to enter valid value, loops until satisfied.	As expected
Main Menu: Input boundary low	Input = 0		Accepts value and continues to next prompt, completes correct number of steps	As expected
Main Menu: Input correct range	0<= Input <= 3		Accepts value and continues to next prompt, completes correct number of steps	As expected
Main Menu: Input too high	Input > 3		Prompt user to enter valid value, loops until satisfied.	As expected

Main Menu: Input boundary high	Input = 3		Accepts value and continues to next prompt, completes correct number of steps	As expected
Crew Quarters: Gate Keeper	If player has match in purse they may enter	Menu.cpp, main.cpp, validator.cpp, random.cpp, BioLab.cpp, ControlRoom.cpp, CrewQuarters.cpp, EngineRoom.cpp, GameController.cpp, PhysicsLab.cpp, Purse.cpp, Room.cpp, Storage.cpp	Player can enter the crew quarters if they have a match and the match is removed from their purse	As expected
Physics Lab: Gate Keeper	If player has jump rope or axe in purse they may enter		Player can enter the Physics Lab if they have a jump rope or axe and the room remains accessible	As expected
Bio Lab: Gate Keeper	If player must enter “75” in purse they may enter		Player is allowed to enter upon providing the code	As expected
Fire Axe: Gate Keeper	If player has ipod in purse they may enter		Player can collect axe if they have the ipod in their purse	As expected
Acid: Gate Keeper	If player has key in purse they may enter		Player can collect acid if they have the key in their purse	As expected
Alien: Gate Keeper	If player has laser, acid, and axe in purse they win		Player win if they have laser, acid, and axe in purse when they find the alien.	As expected
All: Gate Keeper	If play does not have required items		Player is prompted to try again	As expected
Purse holds three items	Player attempts to add fourth item	Menu.cpp, main.cpp, validator.cpp, random.cpp, BioLab.cpp, ControlRoom.cpp, CrewQuarters.cpp, EngineRoom.cpp, GameController.cpp, PhysicsLab.cpp, Purse.cpp, Room.cpp, Storage.cpp	Player is prompted to remove items from purse	As expected
Purse manger allow of the deletion of purse items	Player enters item number to delete		Item is deleted from purse	As expected
Player is allowed 10 min to complete the game	Player takes more than 10 min		Player is prompted: Game over, try again	As expected

Brian Terrell
06/13/2017
Final: Space Station Game

Space Ship Game Design Document - Reflection

This was a great project to do as soon as I got the ball rolling, but it did take a while to get the ball rolling. I spent way too much time in the design phase, which I partially attribute to looking at the example projects and a mental block as to how I would have the spaces interact with the game controller. The examples are awesome in many ways and I found them to be interesting, so I began trying to create a story board that satisfied the design specification while being interesting. This was harder said than done, while I had interesting ideas, I couldn't really get them to fit into the requirements, or the time I had available to complete the project. The second issue in the design phase was where to place the main parts of the game logic. After looking through an example that a had a main.cpp file loaded with room specific code I decided that I would create my game engine to be as simple and generic and let the derived spaces handle as much as possible as it seemed like good practice for the sake of scalability.

I also became a little stuck on how to share the item container Purse.h with all of the derived space classes as well as and how to get player status back out of them. My solution ended up being as simple as pointers, but for some reason I let it become a big deal at first. Once I had a character that could go into a room, pick up an object, and report game status back to the game engine everything else went smoothly. One thing I did to save time was I thoroughly tested/debugged the first derived space class I coded, taking the time to reflect on any possible issues I could have as I constructed the rest of the spaces. I wrote all the comments I planned on making. When I was satisfied that it functioned as designed, I then duplicated and refactored it into the other five derived classes very quickly, with only minor debugging.

For the most part this program was built the way it was designed, I had very few issues with the pointers and no memory leaks that I detected. One change that was made as the program was finished was in how to track and limit time. At first I wanted to keep a tally of how many times the player moved from room-to-room, but then decided that such a system might come off a bit clumsy. Instead I opted for a timer that refreshes every time the main play function loops. This seemed like a better option. Overall this project was great fun to design and code and I think it turned out well.

Raydon,

Thank you for all your hard work this term, you were very responsive every time I had a question. I can only imagine the workload of taking classes, grading, and answering questions.